HW01

September 7, 2021

1 ECE 537: Foundations of Computing

1.1 Homework #1: Hire-Assistant Problem

Write a program in the language of your choice (C++, Python, etc.) that generates all 8! permutations of the array A = < 1, 2, 3, 4, 5, 6, 7, 8 > and computes the number of "hires", H1, H2, ... as described in the lecture. Then compute the theoretical expected value as EH = 1/8!(H1 + H2 + ... + H8!) and the theoretical variance as Var(H) = E(H^2) - (EH)^2 = 1/8!(H1^2 + H2^2 + ... + H81^2) - (EH)^2. Compare the output of your program to the value for EH found in the lecture. Compare also to the value ln (8).

Your submission should contain: (1) the program (2) the values for EH and Var(H) generated by your program (3) the comparisons.

> Note, we adopt the convention that a higher rank corresponds to a better qualified applicant (i.e. 8 being the best candidate, 1 being the worst).

Probability 1/8

H = Number of times we hire a new office assistant

Use indicator random variables to greatly simplify the expected value calculation.

[2]:
```
from itertools import permutations
from math import factorial, log

n = 8
count = 0
A = factorial(int(n))
EH = log(n)
print("There are", A, "permutations.")
print("The expected number of hires is:",EH)

perms = permutations(range(1, n+1))
sum = 0
for k in list(perms):
    count += 1  # Enumerate the permutations
    temp = max(k)   # Find maximum element in each permutation (best candidate␣
 ↪to hire)
```

```
    H = [i+1 for i, j in enumerate(k) if j == temp]  # Use list comprehension to
↪determine the index of the maximum element we just found
    # sum += H[0]
    print(count,": ",k,"\t\tNumber of hires:\t",H[0],"\t")
```

```
There are 40320 permutations.
The expected number of hires is: 2.0794415416798357
1 :  (1, 2, 3, 4, 5, 6, 7, 8)          Number of hires:        8
2 :  (1, 2, 3, 4, 5, 6, 8, 7)          Number of hires:        7
3 :  (1, 2, 3, 4, 5, 7, 6, 8)          Number of hires:        8
4 :  (1, 2, 3, 4, 5, 7, 8, 6)          Number of hires:        7
5 :  (1, 2, 3, 4, 5, 8, 6, 7)          Number of hires:        6
6 :  (1, 2, 3, 4, 5, 8, 7, 6)          Number of hires:        6
7 :  (1, 2, 3, 4, 6, 5, 7, 8)          Number of hires:        8
8 :  (1, 2, 3, 4, 6, 5, 8, 7)          Number of hires:        7
9 :  (1, 2, 3, 4, 6, 7, 5, 8)          Number of hires:        8
10 :  (1, 2, 3, 4, 6, 7, 8, 5)         Number of hires:        7

OUTPUT TRUNCATED TO SAVE SPACE

40310 :  (8, 7, 6, 5, 3, 1, 4, 2)      Number of hires:        1
40311 :  (8, 7, 6, 5, 3, 2, 1, 4)      Number of hires:        1
40312 :  (8, 7, 6, 5, 3, 2, 4, 1)      Number of hires:        1
40313 :  (8, 7, 6, 5, 3, 4, 1, 2)      Number of hires:        1
40314 :  (8, 7, 6, 5, 3, 4, 2, 1)      Number of hires:        1
40315 :  (8, 7, 6, 5, 4, 1, 2, 3)      Number of hires:        1
40316 :  (8, 7, 6, 5, 4, 1, 3, 2)      Number of hires:        1
40317 :  (8, 7, 6, 5, 4, 2, 1, 3)      Number of hires:        1
40318 :  (8, 7, 6, 5, 4, 2, 3, 1)      Number of hires:        1
40319 :  (8, 7, 6, 5, 4, 3, 1, 2)      Number of hires:        1
40320 :  (8, 7, 6, 5, 4, 3, 2, 1)      Number of hires:        1
```

[20]:
```
import numpy as np
#created a bernoulli class

class bernoulli():
    def pmf(x,p):
        """
        probability mass function
        """
        f = p**x*(1-p)**(1-x)
        return f

    def mean(p):
        """
        expected value of bernoulli random variable
        """
```

```
        return p

    def var(p):
        """
        variance of bernoulli random variable
        """
        return p*(1-p)

    def std(p):
        """
        standart deviation of bernoulli random variable
        """
        return bernoulli.var(p)**(1/2)

    def rvs(p,size=1):
        """
        random variates
        """
        rvs = np.array([])
        for i in range(0,size):
            if np.random.rand() <= p:
                a=1
                rvs = np.append(rvs,a)
            else:
                a=0
                rvs = np.append(rvs,a)
        return rvs

p = 1 / 8   # probability of having an accident
print(bernoulli.mean(p), # return -> 0.2
bernoulli.var(p), # return -> 0.16
bernoulli.std(p)) # return -> 0.4
#each execution generates random numbers, so array may be change
bernoulli.rvs(p,size=8)
```

```
0.125 0.109375 0.33071891388307384
```

[20]:
```
array([1., 1., 0., 0., 0., 0., 0., 0.])
```